文件系统的发展脉络

熊劲 潘锋烽 刘凯捷 张子刚 马久跃 姜继 孙凝晖

摘　要:本文回顾了文件系统的发展历程，从计算机技术重大进步的全景考察文件系统的发展脉络，分析现有各种文件系统产生的技术背景，并预测未来的数据中心文件系统可能的技术创新。同时，本文还简要地总结了 20 年来我们在文件系统方面的研究工作。

1　引言

数据是信息的载体。文件系统是持久保存数据和管理数据的一种最普遍、最基本的手段。随着应用需求的变化和计算机技术的进步，文件系统也逐渐演变发展，形成了多种不同类型。它们有各自的应用场景和不同的特点。按照它们出现的先后顺序，文件系统可划分为单机文件系统、网络文件系统、并行文件系统和分布式文件系统[1]。值得强调的是，这些不同类型的文件系统之间并不是一种进化关系，不是后一种文件系统出现就意味着前一种文件系统就消亡了。它们共同存在、共同发展，分别服务于不同的应用需求。

在当今这个信息爆炸时代，数据变得越来越重要。越来越多的应用需要处理越来越庞大的数据集。而且，科学与工程计算、Web2.0，以及近年热起来的云计算、虚拟化、移动终端等，都加剧和加速了数据量的膨胀。在新的应用需求下，业界掀起了开发文件系统的热潮。很多互联网公司和存储公司都在开发自己的文件系统，甚至一些应用单位也在根据自己应用的特点开发专用文件系统。在这种形势下，人们自然要问：现有文件系统在哪里出了问题？未来的文件系统应该是什么样的？为了探讨这些问题，我们试图从计算机技术重大进步的全景来考察文件系统的发展脉络，分析现有文件系统产生的背景和未来可能的技术创新。

2　单机文件系统

文件系统是操作系统用来管理文件和存储空间的子系统，因此它是操作系统必不可少的一部分。正由于这个因素，文件系统是伴随着操作系统的产生而逐渐出现的。汉森（P. Hansen）总结了操作系统的主要发展阶段和每个阶段的主要技术创新[1]，如表 1 所示。

现代文件系统的起源要追溯到分时操作系统时期。在分时操作系统出现之前，计算机采用批处理方式，把一批作业以脱机方式输入到磁带上，在监督程序的控制下一个作业接一个作业地连续处理。作业和数据通过穿孔卡片输入计算机，永久保存在磁带或磁鼓等外围存储设备上。这种方式存在交互性差以及作业周转时间长等问题。为了满足用户人机交互、共享主机、便于用户上机的需求，再加上磁盘技术在当时的发展，研究者们提出了分时操作系统（Time Sharing Operating System），其中较为著名的就是 Multics。

Multics 是 1965 年左右由美国电话电报公司（AT&T）贝尔电话实验室、通用电气公司、麻省理工学院 MAC 课题组一起联合开发的一个多用途（General-Purpose）、分时（Time-Sharing）及多用户（Multi-User）的操作系统[2]。其目标是要能够支持众多用户同时使用计算机，提供强大的数据存储，以及允许用户能够容易地共享他们的数据。开发者们发

[1] 本文的分布式文件系统特指服务于互联网应用海量数据存储需求的分布式文件系统。

表1. 操作系统的主要发展阶段及其主要技术创新 [1]

主要阶段	操作系统	技术创新
开放式计算站（Open Shop）	IBM 701 open shop(1954)	操作系统的概念
批处理	BKS system(1961)	磁带批处理、先入先出调度
多程序设计	Atlas supervisor(1961) B5000 system(1964) Exec II system(1966) Egdon system(1966)	处理器复用、原子操作(Indivisible operations)、按需分页、读写外部设备联机并行操作（Input/output spooling）、优先级调度、远程作业输入
分时系统	CTSS(1962) Multics file system(1965) Titan file system(1972) UNIX(1974)	即时用户交互（Simultaneous user interaction）、在线文件系统
并发编程	THE system(1968) RC 4000 system(1969) Venus system(1972) Boss 2 system(1975) Solo system(1976) Solo program text(1976)	层次结构系统、可扩展内核（Extensible kernels）、并行程序设计概念、安全并行语言（Secure parallel languages）
个人计算	OS 6(1972) Alto system(1979) Pilot system(1980) Star user interface(1982)	图形用户界面（GUI）
分布式系统	WFS file server(1979) Unix United RPC(1982) Unix United system(1982) Amoeba system(1990)	远程服务器

表了一系列有关于 Multics 的文章，其中一篇就描述了 Multics 文件系统所需要的特性和结构[3]。在这篇文章中首次提出使用树型结构，而不是使用之前的平面结构（例如 DECtape）来组织来文件、目录的思想，同时也提出了对于文件和目录的访问控制，从而奠定了现代文件系统的发展基础。

2.1 Unix 文件系统和 FFS

Multics 项目中很多重要的思想和设计理念影响着当时参与这个项目的贝尔实验室的两位软件工程师 汤普逊（Ken Thompson）与瑞奇（Dennis M.Ritche）。1969 年，在他们为 DEC PDP-7 开发操作环境 UNICS 时，借鉴了很多 Multics 的设计思想。后来，这个系统被称为 UNIX。当时的 UNICS 包含两大功能：（1）一个简单的文件系统，即 PDP-7 file system[4]，它后来发展成为 UNIX 文件系统的早期版；（2）一个进程子系统和 shell（命令解释器）。其中的 PDP-7 file system 借鉴了引文[3]中的很多思想，而且它使用了 i-节点（i-node）来描述一个文件，有特殊的文件类型来支持目录和设备[4]，可以同时支持两个用户，这已经和现代文件系统很相似了。

1973 年，汤普逊和瑞奇使用 C 语言对 UNIX 进行了再加工和编写，使得 UNIX 能够很容易地移植到其他计算机上。之后，他们发表了首篇 UNIX 论文——《UNIX 分时系统（*The*

UNIX Time Sharing System）》[5]。这篇文章不但奠定了现代 UNIX 操作系统的结构，而且首次提出以“流”的方式来实现文件的访问，提供了极大的灵活性，从此奠定了文件系统的基本结构、功能和接口。而这时的文件系统一般称为 UNIX 文件系统。

UNIX 文件系统是现代文件系统的一个基础模型。从结构上看，它是从 PDP-7 文件系统衍化而来的。它四个主要模块是引导块、超级块、i-节点、数据块，这些都被运用到了之后的文件系统中。而且它的设计简单，大大减少了工作量。但是它也有很多缺点：磁盘分配的单位为 512 字节，系统的性能低下，吞吐量只有磁盘带宽的 2%—5%，读写（I/O）时间过长，而且不够可靠等。虽然 BSD[2]在 1978 年设计了 `fsck`[3]来修复文件系统，通过把磁盘块大小增加到 1KB，将性能提升了 2 倍，但仍然没有彻底解决性能问题。

1984 年，加州大学伯克利分校的研究者提出了快速文件系统 FFS[6]。它是最早注重于读写性能优化的文件系统，也可以认为是现代文件系统的鼻祖。它提出的很多思想、方法至今仍在使用。

为了提高读写性能，FFS 将磁盘分配单位大幅度提高到 4KB，磁盘带宽的利用达到了 14%—47%；同时为了减少碎片而导致的空间浪费，它又提出了分片（fragment）方法，将一个数据块（block）分割成多个固定大小的片，小文件以片为粒度分配空间，从而节省了空间。

为了减少磁头寻道时间，FFS 将磁盘划分为若干个柱面组（cylinder group），每个柱面组由若干连续的柱面组成，其中包含各自独立的超级块、i-节点区和数据块。FFS 可以将相关的数据存在同一柱面组中，从而使磁头移动距离最小。其他改进还有长文件名、符号链接等。

2.2 单机文件系统的发展

2.2.1 日志结构文件系统 LFS

1992 年， LFS（Log-structured File System，日志结构文件系统）[7]第一次提出将文件系统作为日志来实现。日志结构文件系统主要解决磁盘在大量小粒度随机写负载时性能差的问题。它定义了以 segment（“段”，由连续的磁盘块组成）为基本的磁盘存取单位，以“添加到已存在的日志（append to log）”的方式将数据写入到磁盘上，这样的好处是只需要检查最后写入磁盘的内容就可以处理机器崩溃等异常。但是同时带来的问题是，一旦日志的空间用完了，就需要进行 segment 清理。这个阶段会占用绝大部分的磁盘带宽，从而影响 LFS 的性能。另外， LFS 虽然将很多小粒度的写请求聚集成为一个 segment，让其性能大大提升，但是读取的时候数据是分散在各个 segment 中的，会造成对磁盘小粒度的随机读，因而读性能受很大影响。因此 LFS 的思想和方法在磁盘文件系统中未得到普遍使用。直到 NAND 型的闪存（Flash）存储介质出现后，很多闪存文件系统都借鉴了 LFS。

2.2.2 可扩展文件系统 XFS

XFS[9]是 SGI 公司 1994 开发出来文件系统。在它之前的文件系统都采用类似 FFS 的技术，文件系统大小不能超过 4-8GB，文件长不能超过 2-4GB。而且目录采用线性组织，文件数量多时，查找效率很低。而当时的视频播放应用需要很高的读写带宽和很大的存储空间。例如，当时的 VOD 服务器保存 1000 部影片需要 2.7TB 的磁盘空间，同时支持 200 个 MPEG 流需要 100MB/s 的 I/O 带宽（每个 MPEG 流的速度是 4Mbps，200 个并发 MPEG 流的总速

[2] Berkeley Software Distribution，伯克利软件套件,是 Unix 的衍生系统
[3] UNIX 及与其类似的操作系统上的一个工具，用以检查文件系统的一致性

度是 800Mbps）。

XFS 主要创新在于突破了以前文件系统在存储容量、文件大小、文件数量等方面的限制，主要通过将数据库系统中的 B+树索引技术和日志技术引入到文件系统中。XFS 是 64 位的日志文件系统，因此可以管理极大的磁盘空间和数据，也允许意外重启后的快速恢复，增强了其可靠性。在设计时，XFS 充分考虑了系统的可扩展性，大量采用 B+树结构（原因在于 B+ 树的优越性能和极好的可扩展性）来组织管理如目录这样的数据；结合 B+树和“变长单元（extent）”管理磁盘空间、文件数据块等；基于文件的方式管理 i-节点，因此文件系统可以动态扩展。

XFS 使用缓存技术来提高每次写入磁盘的数据量（延迟高达 30s），以减少磁盘寻道时间并提高系统吞吐量。同时引入了“分配组”的概念，以便能有效地处理并行读写。

由于 XFS 的高性能、良好的伸缩性、健壮性，迄今仍在大量使用，很多存储产品都是基于该文件系统。

2.2.3 ZB 级文件系统 ZFS

2004 年，Sun 公司发布了 ZFS[10]。ZFS 既是几十年来文件系统优秀技术的集大成者，更是一次对已有技术彻底的颠覆和重生。它从根本上改变了文件系统的管理方式。其主要特征包括：

ZFS 设计了全新的文件系统结构，高效地整合了过去文件系统的“卷管理”功能和 RAID[4] 功能，提出“存储池”的概念来管理物理存储空间，极大地简化了系统管理，而文件系统结构本身却变得更加清晰、精简。ZFS 能够自动检测并修改文件数据的损坏；对文件系统的元数据实现冗余保护；采用“写时复制事务模型（Copy-on-write Transactional Model）”来维护数据一致性，能够高效地创建快照，为数据提供了最全面的保护。而且 ZFS 是一个 128 位的文件系统，可以说在相当长的未来时间内，ZFS 几乎不太可能出现存储空间不足的问题。因此，ZFS 在当时又被 Sun 称为是终极文件系统。

2.3 负载特征和性能评价

单机文件系统的负载是多进程并发访问和交互式访问。因此，单机文件系统的性能评价标准包括聚合读写带宽和读写请求响应时间。

3 网络文件系统

早期的大型机分时系统配合终端的使用方式就可以认为是一种网络，节点分布在不同物理位置。这种早期的网络连接基于电路交换技术，带宽利用率低且可靠性差。直到分组交换技术的出现，才使得组建可靠的大规模网络成为可能。

在 20 世纪 70 年代和 80 年代，出现了剑桥环、令牌环、星形网络等各种实验性和商业化的局域网技术。1974 年为了互连各种不同网络而由美国国防部高等研究计划局（DARPA，Defense Advanced Research Project Agency）主导提出了 TCP/IP 网络协议[12]。1983 年 ARPANET[13]全部转入 TCP/IP 协议，成为互联网的雏形。而在局域网的技术方面，施乐公司帕洛阿尔托研究中心（Xerox PARC）发明的以太网技术由于技术优秀以及大厂商的推广逐渐成为工业标准。局域网在各种应用场合开始得到广泛使用。

[4] Redundant Array of Inexpensive Disks ，冗余磁盘阵列

3.1 早期的网络文件系统

施乐帕洛阿尔托研究中心先进的实验室环境由 Alto 个人电脑和以太网组成，并且先于 TCP/IP 的标准制定开发了 PUP（PARC Universal Packet）网络协议[14]用在实验室的以太网环境。在开发网络文件系统之前，帕洛阿尔托研究中心使用的是运行 Tenex 系统[15]的大型机作为文件服务器，使用基于 PUP 协议的 FTP[5]服务来访问文件。随着文件规模的增大，为了方便研究人员的工作，帕洛阿尔托研究中心的研究人员开发了最早的网络文件系统之一 IFS[16]来满足需求。

IFS 和后来的 XDFS[17]、以及剑桥大学的 CFS[17]等，这一类的分布式系统一般被称为文件服务器（File Server）而非文件系统（File System）。IFS 提供了网络环境下客户端访问服务端文件的各类接口，其实现基于 FTP，并具有备份、归档等管理功能，但在这类系统中访问文件的接口和访问本地文件的方式是不同的，也就是不具有网络透明性[18]。

Newcastle Connections[19]是纽卡斯尔大学开发的网络文件系统，运行在 PDP-11 的 UNIX 操作系统环境，提供具备网络透明性的文件系统方案。Newcastle Connection 系统将剑桥环局域网中的 UNIX 主机组成一个共享的名字空间，文件在共享名字空间中由主机名的前缀和本地路径标识。Newcastle Connection 中提供了类似本地文件系统访问文件的方式来访问分布式环境中的文件，但名字空间的设计决定了其文件名字与物理地址相绑定，不具备位置透明性。此后的 Locus 系统提供了一个具有位置透明性的名字空间，使得分布式系统的使用和管理更加方便。

3.2 NFS 和 AFS

TCP/IP 和以太网的出现，使得局域网在各种应用场合开始得到广泛使用。在局域网环境中，用户和数据分散在网络中的各个机器上，用户迫切需要在不同机器之间共享数据。最初的解决方案是将整个文件从一个机器复制到另一个机器上，比如 UUCP[6][20]和 FTP[21]。这种方法对于部分需求来说简单可行，但不够理想。比如，数据在多个机器上冗余存储，维护这些冗余数据的最新版本也不方便。为了让用户能够以访问本地文件系统的方式来访问远程机器上的文件，各种类型的网络文件系统应运而生。

在 1975-1985 年间产生了许多不同设计风格的网络文件系统。但最终得到广泛使用的主要是两种：NFS[7][22, 24]和 AFS[8]/DFS[23]。它们并没有全面超越同时代的其它系统，但因为其突出的优点而适用于各自的使用环境。

(1). NFS

NFS 由 Sun 公司在 1984 年开发。NFS 使用 IP 或 UDP 协议传输，被认为是第一个得到广泛应用的现代网络文件系统。NFS 的核心设计目标是提供跨平台的文件共享系统。它使用平台无关的 XDR[9][25]数据描述编码的协议，文件操作通过 RPC[10][26]机制实现。除此之外为了实现用户使用的透明性，Sun 公司在 4.2BSD UNIX 内核基础上实现了 VFS[11]层，使得 NFS 用户可以用几乎完全相同的方式访问本地和远程文件系统。NFS 的协议被纳入 RFC 标准，

[5] File Transfer Protocol，文件传输协议
[6] Unix-to-Unix Copy
[7] Network File System，网络文件系统
[8] Andrew File System，安德鲁文件系统
[9] External Data Representation Standard，外部数据表示标准
[10] Remote Procedure Call Protocol，远程过程调用协议
[11] Virtual File System，虚拟文件系统

并且其实现和设计思想都相对简单，容易进行性能调优。所以 NFS 被广泛接受，在产业界和学术界都获得了大量应用。

NFS 使用无状态协议，优点是实现简单而且错误恢复容易，缺点是传输冗余指令多，以及无法支持完整的 UNIX 文件系统语义，比如文件加锁等。除此之外，NFS 单一服务器的结构也决定了它的扩展性有限。

(2). AFS 与 DCE/DFS

AFS 是卡内基梅隆大学 1982 年开始开发的分布式文件系统。其设计目标是支持 5000-10000 个节点的集群，扩展性是首要考虑因素。首先与 NFS 等系统不同的是，AFS 集群中有专用服务器，并且为了防止服务器的 CPU 资源成为瓶颈，AFS 系统部分计算任务由客户端分担。因此，AFS 比 NFS 具有更好的扩展性。

AFS 中有多个服务器，整个名字空间被较静态地划分到各个服务器上。一组服务器和一组客户端形成一个小的集群，各个集群用主干局域网（backbone LAN）互连。一般来说系统避免跨集群的操作以保证性能。AFS 为用户提供统一的名字空间，并以卷为单位进行管理、迁移，卷与物理服务器的位置映射保持在一个数据库中并在所有服务器均有备份。同时为了降低服务器计算负担，AFS 的客户端在本地磁盘缓存整个文件。缓存的策略使得 AFS 受限于会话级别（一个会话是从打开文件开始，到关闭文件为止）的共享语义支持，并且缓存一致性的保证机制增加了系统复杂度，同时本地磁盘的缓存使用使得 AFS 不能像 NFS 那样很好地支持无盘工作站。

1989 年 Transarc 公司接管 AFS 的开发和产品化工作，最终研制出的系统被称为 DCE[12]/DFS[13]。DFS 从 AFS 演化而来，做了以下几个方面的改进[27]：

1. DFS 中，一台机器既可以是客户机又可以是服务器。
2. DFS 提供了类似 UNIX 的共享语义和一致性保证机制（读写操作级别）。
3. DFS 可以跟其他文件系统有更好的互操作性。

但 DFS 体系结构很复杂，不仅需要 DCE RPC，而且需要 X.500 全局目录服务等相关服务，在小的机器和简单的操作系统上很难支持 DCE/DFS。

3.3 负载特征和性能评价

网络文件系统的负载是多用户共享文件访问，通常是各个用户从不同的客户端访问各自的文件，也存在多用户对同一文件的共享访问，不过以共享读为主。用户对单个读写访问的性能要求并不高，但由于需要支持多个用户并发访问，因此，要求比较高的聚合读写带宽。网络文件系统的性能评价标准以聚合读写带宽为主。

3.4 网络存储

随着存储容量的不断增大，以及用户对于数据安全性和可靠性需求的提高，存储设备逐渐从处理业务逻辑的主机中分离出来，出现了 JBOD[14]、RAID 磁盘阵列等设备，提供高性能、高可靠的存储服务。但以传统的 SCSI[15]线缆连接存储设备终究无法满足快速增长的扩展性要求，同时多种多样的存储设备给整个信息技术设施管理带来很大的复杂性，于是出现了

---

12 Distributed Computing Environment，分布式计算环境
13 Distributed File System，分布式文件系统（此处并非指 Windows 的分布式文件系统 DFS）
14 Just a Bunch Of Disks，磁盘簇,在一个底板上安装的带有多个磁盘驱动器的存储设备
15 Small Computer System Interface，小型计算机系统接口

网络化的存储架构[28]。网络化存储的软硬件主要需解决存储的集中管理和扩展性两个问题，做到在满足大容量存储需求的同时节约成本、方便管理。

施乐帕洛阿尔托研究中心构建的以太网环境中，已经有了 Alto 用作服务器的文件共享概念。之后的 3Com 公司大力推广以太网的同时开发了运行在 DOS 系统的 3+Share 网络文件共享服务。同时 3Com 公司为 3+Share 开发制造了专用服务器 3Server。3Server 使用 X86 架构和 DOS 操作系统，但没有对键盘，显示器等设备的支持，而它拥有 7 个磁盘插口，以及相应的管理软件，专为提供存储共享服务而设计制造。这种专用服务器硬件的设计思路对存储设备的发展产生了很大影响。在此之后，Auspex 等公司也开始为 UNIX 开发制造支持 NFS 等开放标准的文件服务器，通过专门设计的硬件和定制的操作系统来达到很高的文件服务性能，这样就出现了称为 NAS[16]的存储架构[29]。

NAS 指的是在以太网环境中，瘦客户端连接后端存储服务器的客户/服务器（Client/Server）存储架构，利于在成本经济的同时提供高性能的存储服务。解决客户端和存储服务器文件级交互的就是 NAS 网络文件系统。典型的是在 UNIX/LINUX 上使用的 NFS 和 Windows 上使用的 CIFS。NAS 网络文件系统同时运行在客户端和服务器上，通过以太网传输某种协议的 VFS（虚拟文件系统）层指令，表现为在客户端 RPC 调用服务器的文件系统的访问功能。对于 NAS 系统来说，显著特点是将文件系统的管理集中到了服务器端。

NAS 主要是集中管理存储设备，并提供文件共享服务。从存储设备的扩展性角度来说，光纤通道（Fibre Channel）技术的发展使得通过一个高速、可靠的网络来连接更大量的磁盘和磁带等设备成为可能。也就是说，用光纤通道取代 SCSI 线缆，在存储服务器的读写总线与存储设备之间通过一个光纤通道网络进行连接，这就是 SAN[17]存储架构[30]。

SAN 架构将多种存储设备形成一个统一访问、统一管理的存储池网络。与文件级别的 NAS 不同，SAN 通过高速光纤通道网络进行的是块级的数据访问。由于避免了 NAS 使用以太网与文件系统层通信的开销，并且在块级传输中使用光纤通道高速网络，SAN 架构可以提供较高的读写性能，但 NAS 架构也有其明显的优势，（1）不需要光纤通道适配卡和交换机，成本大大降低；（2）基于 TCP/IP，扩展性很强；（3）可以实现多个客户端的共享访问，而文件系统分离在各个客户端上的 SAN 架构则做不到对底层存储的共享。

SAN 有其性能上的优势，而 NAS 则有低成本和易于部署的优势。为了共享访问存储，SAN 架构中必须有集中管理的系统。一种解决方案是在 SAN 之上搭建 NAS 架构，通过 NAS 网关来访问底层 SAN 架构的存储。另一种办法是使用专门的 SAN 集群管理系统，也就是 SAN 文件系统。

SAN 文件系统提供了共享访问 SAN 存储网络的平台，主要解决的是在文件元数据统一管理的问题。元数据管理的方式有多种，主要分为以 SANergy、CXFS、StorNext 等为代表的集中式元数据管理方式和以 StorageTank 为代表的分布式元数据管理方式。SAN 文件系统客户端获得元数据信息后，可以通过高速光纤通道网络直接存取 SAN 中的块数据，而不必经过 NAS 服务器，使 SAN 架构较高的读写性能得以充分发挥。

4 并行文件系统

20 世纪 80 年代后期，石油勘探、核爆炸模拟等大规模科学计算需要越来越高的计算能

16 Network Attached Storage，网络附连存储（也有译作“网络附接存储”）
17 Storage Area Storage，存储域网（或存域网）

力，而当时的计算机还是单处理器，计算能力有限，不能满足科学计算的需求。因此，出现了以追求高计算能力的多处理器结构的计算机。典型的多处理器计算机采用大规模并行处理（Massive parallel Processing， MPP）体系结构，科学计算应用采用并行编程模型来获得高的计算速度。因此，每个任务由多个协同工作的进程组成，每个进程可以运行在不同的节点上。并行程序对数据访问提出了新的需求，即运行于多个计算节点上的进程共享一个全局文件系统视图，而且需要与高计算速度相匹配的读写性能，由此引发并行文件系统研究。

早期的并行文件系统有 Bridge File System [31]、Concurrent File System（CFS）[32]等。它们运行在 MPP 结构的超级计算机上。由于科学计算的数据文件通常非常大，并且每次访问数据的粒度很大（一般为几 KB 到几十 KB），并行文件系统将文件条带化，分散地存储在所有读写节点。因此，它们与网络文件系统、分布式文件系统的最大不同在于，通过条带化存储，一个读写请求可以并行地发给多个读写节点，这些读写节点并行进行磁盘访问，从而可以为应用提供很高的并行读写带宽。

CFS 等虽然将数据分散到多个存储节点上，能够满足并发访问的要求，但是文件系统向应用程序隐藏数据划分信息。这样，文件系统不能根据应用程序访问模式来存放数据。Vesta 并行文件系统[34]针对这个问题，提出 cell（单元）概念，引入二维数据存储结构；cell 可以看做虚拟读写节点。

随着处理器和网络的性能不断提高和价格的日益下降，使得并行计算逐渐从 MPP 结构的超级计算机向机群结构的高性能计算机转移。机群结构的高性能计算机由一组高性能节点（工作站或者服务器）构成，进而引发基于机群结构的并行文件系统的研究，典型代表有 PVFS[35]、Lustre[36]等，它们运行在当今很多机群结构的高性能计算机上。

4.1 MPP 上的并行文件系统

20 世纪 80 年代末到 90 年代末，大规模并行处理巨型机（MPP）是当时高性能科学计算的主要硬件平台。MPP 由大量计算节点和读写节点组成，通过专门的高速互联网络将这些节点连接起来，提供强大的计算能力和通信能力。计算节点用来运行用户任务，读写节点运行文件系统的服务进程，执行对文件块的读/写请求。

CFS[32, 33]是第一个将大文件分散到多个读写节点以提高读写性能的商业化并行文件系统。它是 20 世纪 80 年代末英特尔（Intel）为其超级计算机研制的并行文件系统，运行于 MPP 结构的超级计算机 Intel iPSC/2[18]上。

CFS 中大文件采用条带化方式存储，一个大文件被划分为大小为 4KB 的块，这些文件块以轮转（round-robin）的方式存储在所有可用读写节点上，如图 1 所示。例如，系统有 N 个读写节点，块 i 放置在第 k 个读写节点上，k=i mod N。所有的读写节点上都运行 CFS 的 `disk`（盘读写）进程，它的主要功能是接收来自计算节点的文件块读写请求，执行真正的磁盘数据访问。CFS 的 `name` 进程只运行在一个读写节点上，它的主要功能是管理目录树，服务所有涉及目录的读写请求。运行于计算节点上的应用程序通过

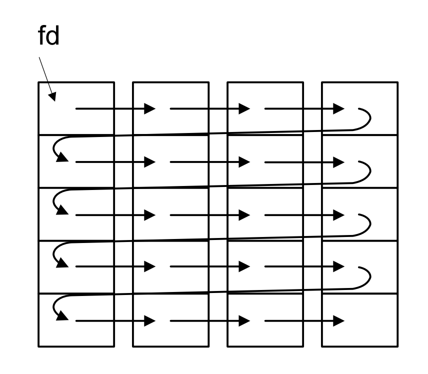


图1. 文件条带化存储 [34]

[18] iPSC/2 是英特尔研制的超级计算机

内嵌的 CFS 实时读写库来访问 CFS 文件。

并行文件系统引入条带化存储方式，将文件划分为等长的单元并循环放置到多个磁盘上，支持对磁盘的并行访问，但是向应用程序隐藏了数据分布信息。这种方式的一个弊端是应用程序开发者无法根据应用程序特定访问模式来划分数据进行存储，因此无法获得更高的访问效率。

为此，Vesta 并行文件系统[34]引入新的文件抽象概念，应用程序开发者可以根据应用程序数据访问模式来指定文件划分模式。Vesta 允许用户将文件划分为多个互不覆盖的块，这些块可以被并行访问。文件块划分可以动态改变，减少文件访问对同步和一致性需要。

Vesta 文件系统有如下两个创新：第一，它不依赖系统中读写节点数；第二，它允许文件采取多种划分方式，可以根据应用程序不同进程的访问模式，将数据划分成不同的块。

Vesta 不依赖系统中读写节点数，使用了 cell 概念，可以把 cell 看做数据容器，或者虚拟读写节点。这些虚拟读写节点可以映射到实际物理节点上。当创建文件时，给定它使用的 cell 个数参数。如果 cell 数目不多于读写节点数，则每个 cell 分布在不同的读写节点上；如果 cell 数目多于读写节点数，则将 cell 循环放置到所有读写节点上。为了获得最佳性能，最好每个 I/O 节点分配相同数目的 cell。

cell 的出现，使得 Vesta 引入二维数据结构。第一维是 cell（图 1 中的水平维度），指定数据访问并发度。第二维是 cell 内的维度（竖直维度）。cell 由一些基本划分单元（Basic Striping Unit，BSU）组成。BSU 可以是任意长度，并且反映数据访问最小单元。cell 数目和 BSU 大小，是定义文件的两个参数。文件创建时被指定，以后不能修改。

4.2 Linux 机群上的并行文件系统

20 世纪 90 年代后期，并行计算逐渐从 MPP 结构的超级计算机向由机群结构的高性能计算机转移，机群成为构建可扩展并行计算机的主流方式，主要原因包括以下几个方面。

（1） 处理器速度随时间呈指数增长，同时价格日益下降[37]。因此，在 20 世纪 90 年代后，处理器的速度已经非常快，利用商品化的工作站就可以获得很高的计算速度。

（2） 局域网上高速网络技术和通信协议不断完善，节点间通信能获得更高的带宽和更小的延迟。互连网络是并行计算机的关键部分。早期的并行计算机使用专门的互连网络，典型代表有 SGI 的 NumaLink、IBM 的 SP Switch、SP Switch2、克雷（Cray）的 Cray Interconnect 等。随着高速互连网络技术的不断发展，新的互连技术不断出现，如 Myrinet、QsNet、千兆以太网，Infiniband 等。

（3） 由于机群系统采用包括服务器、高速互连网络、磁盘阵列等商品化部件组成，因此机群系统比传统的并行计算机更易于构造，而且性价比也更高。

（4） 机群上的开发工具日趋成熟，而传统的并行计算机上缺乏一个统一的标准；

（5） 机群还具有良好的扩展性，通过增加节点内存或者更换 CPU 便可获得更高性能。

机群是由一组独立的计算机（节点）组成，节点间通过高性能的互联网络连接。节点都有自己的本地磁盘和完整的操作系统。互连网络通常使用商品化网络，如以太网、Myrinet、Infiniband 等。节点间以松耦合的方式相互连接。节点除了可以作为一个单一的计算资源供用户使用外，还可以协同地工作，像一个单一的、集中的计算资源那样完成并行计算任务。

20 世纪 90 年代中期开源的 Linux 操作系统逐渐成熟并得到广泛的使用，很多有高性能计算需求的实验室通常采用计算机机群方式来构建高性能计算环境，Linux 机群越来越普遍。而上述并行文件系统则需要运行于特定厂商生产的 MPP 结构的超级计算机上，无法运行于越来越多的 Linux 机群上。于是，人们开始开发运行于 Linux 机群的并行文件系统，典型代表包括 PVFS 和 Lustre。它们都是开源软件，因而被广泛地使用，这也促进了它们自身的发展。

PVFS 和 Lustre 吸收了 MPP 并行文件系统的很多思想，包括：（1）采用一个专门的服务器（元数据服务器）来维护和管理整个文件系统的名字空间，为所有计算节点上的应用和用户提供一个全局的名字空间视图；(2)将文件数据条带化并分散存储在所有的读写节点（存储服务器）上，并且提供并行读写接口[40]。这样，一个读写请求可以并行分发到多个读写节点上，从而获得很高的并行读写带宽；（3）它们还利用高速互连网络（Myrinet、InfiniBand 等）的底层通信协议来传输文件数据，从而能够更好地利用高速互连网络的带宽提供更高的读写性能。

除此之外，PVFS 和 Lustre 还提供了一些 MPP 并行文件系统所没有的功能，包括：（1）支持传统 POSIX 标准的文件访问接口。由于它们的客户端除了并行读写库外，还实现了 VFS 层接口，从而应用程序和用户可以直接使用操作系统提供的系统调用、库函数和系统命令来访问并行文件系统。这样使得它们所能够支持的应用范围不再局限于并行程序，还包括串行程序和使用本地文件访问接口的程序。（2）读写节点可以与计算节点重合。PVFS 和 Lustre 中，文件系统的客户端与服务器端的划分是逻辑上的，物理上这些组件可以重叠放置在同一个物理节点上。这样它们能够支持更灵活的机群架构。

4.3 负载特征和性能评价

对于高性能计算应用，一个并行作业是由多个分布于不同计算节点的任务构成的，因此，并行文件系统的负载主要有两类：（1）多个任务对同一文件的不同位置的并行访问，即并行文件访问接口[38]；（2）一个读写请求由多个存储服务器和多个磁盘并行工作来满足。这两类并行 I/O 负载都强调高带宽，因此，并行文件系统的性能评价标准以并行读写带宽为主。

5 分布式文件系统

20 世纪 90 年代后期，随着互联网的发展，无论是用户数量还是网页数量都随时间呈现指数增长。由于网页数量太大，人们需要一种工具来帮助自己从互联网上快速找到所需要的网页，于是出现了海量文本检索工具，即搜索引擎。搜索引擎需要把整个互联网上的网页都抓取回来，对它们进行分析处理（如去重、分词、计算排序（PageRank）等），并构建倒排索引，以便支持用户快速检索到自己想要的网页。建索引是一个海量数据处理的应用，它每次都需要对整个互联网的所有网页进行处理，无论是数据量还是计算量都非常大。而且，为及时反映互联网上新增的网页，抓取网页、分析处理、建立索引这个过程是反复进行。不过，分析处理和建索引是以读为主，而且容易并行。在查询方面，当时的搜索引擎每秒服务数千个请求，而每个请求需要数百亿的处理器周期，需要读取数百兆字节的数据[39]。搜索引擎需要系统具有高吞吐率、低成本，而不是高峰值处理性能。于是产生了以谷歌（Google）的 PC-cluster [39]、Google File System (GFS) [40]、MapReduce [41]为代表的新型数据处理架构。

实际上在 GFS 出现之前，高性能计算（HPC）领域提出了集群上的并行文件系统的解决方案。这些方案通过将数据分片并存储在不同的节点，由并行访问来提高读写性能。同时使用高可靠的服务器级存储设备、磁盘冗余阵列以及检查点等技术来保证数据存储的可用

性。并行文件系统方案能够很好地满足高性能计算领域的高性能读写需求，但是对于互联网应用的海量数据处理需求，它们却不适合。主要原因在于：首先，为了满足峰值读写带宽的需求和数据可靠性的需求，高性能计算领域中使用价格昂贵的高端磁盘阵列。在当时，单个高性能计算应用处理的数据量远远低于搜索引擎处理的数据量。对于互联网应用的海量数据处理需求，这种解决方案成本太高，且性价比低。其次，将数据可用性保障完全交由硬件也是不现实的，即使能够保证存储设备是可靠的，但如果网络或计算机本身发生故障，相应的数据也处于不可用的状态。第三，存储设备也有不可用的时候，RAID 技术虽然能够容忍磁盘故障，RAID 恢复时不仅对性能造成很大的影响，同时重建过程会增强其余磁盘的读写压力，使得其余磁盘发生故障的可能性增大，有可能会造成数据的永久丢失。虽然使用检查点技术能够将数据恢复到上一个一致状态，但需要对系统进行停机操作，这对于需要 365×24 在线服务的互联网应用显然是不适合的，而且大部分互联网应用无法进行“重做（Redo）”操作，丢失的数据无法进行恢复。

5.1 GFS

GFS 是在搜索引擎需要对海量数据进行存储和处理这样的需求下产生的。为了解决海量数据处理问题，谷歌提出了基于廉价计算机集群的分布式数据存储与处理架构，使用廉价的 PC 集群取代昂贵的服务器集群来降低成本。由于互联网应用中的请求大部分都只需要简单的计算，但请求数据量非常大，因此使用高端服务器是一种资源浪费，而 PC 集群，由于成本低，可以通过数量上的优势更好地并行处理海量的请求。因此使用 PC 集群是性价比最优的一种解决方案。但由于 PC 机发生故障的概率要远高于服务器，且集群规模十分庞大，所以在硬件上不可能保证数据的可用性。同时软件缺陷、人为操作错误或者是网络电力的失效都有可能造成结点失效，而且失效是经常发生的，所以需要在软件层中实现数据的可用性保证。

GFS 底层平台是大规模（数千台到数万台）的、廉价的、可靠性较低的 PC 集群，存储设备是集群中每个节点上的多块 IDE[19]磁盘。搜索引擎的负载具有特殊性，它操作的文件大小一般都是 GB 级，而且对于这些文件的操作大部分是只读或追加操作，没有随机写操作，因此，GFS 的设计目标是为 GB 以上的大文件提供高的读写访问带宽，而不是低访问延迟。根据应用的需要，GFS 重新设计了文件访问的 API[20]接口及语义，而没有支持传统文件系统的 POSIX 接口及语义。GFS 采用集中式管理、分布式存储这种架构。由一个单一的主节点（Master）管理整个文件系统的元数据，数据分布存储在集群中大量的节点上，这些节点称为 Chunkserver。每个文件划分为 64MB 粒度的大块（称为 Chunk），存储在不同的节点上。主节点不仅维护整个文件系统的目录和文件，而且还维护每个文件的每个块的存储位置信息，以及每个 Chunkserver 的状态（是否激活）信息。出于性能考虑，主节点在内存中维护所有这些信息。为保障数据的可靠性和可用性，GFS 中每个数据块都保存多份复本（缺省是保存 3 份）。每个写操作都同时写多个复本，只保证多份复本的弱一致性。为避免主节点成为单一故障点，每个 GFS 都有 2 个本地的备份主节点和 1 个异地的备份主节点。

在谷歌公司使用上述架构取得成功之后，其它互联网公司由于自己数据量的增长也面临海量数据处理的问题，很多也采用与谷歌类似的技术来解决数据规模的问题。比如，雅虎大力支持开发并采用了 Hadoop[42]，它是 GFS 和 MapReduce 的一种开源实现。而且 Hadoop 还被 Facebook 等很多企业所采用。微软开发了它的数据中心文件系统 TidyFS[43]和它的并行数据处理系统 Dryad[44]。TidyFS 和 Dryad 在本质上分别与 GFS 和 MapReduce 相似。

---

19 Integrated Device Electronics，电子集成驱动器
20 Application Programming Interfaces，应用程序接口

5.2 与应用融合的专用文件系统

正如前面所述，GFS 的设计目标是为 GB 以上的大文件提供高的读写访问带宽。而在它之后出现的一些新应用却需要对几 KB 到几 MB 的小文件提供低延迟、高吞吐率的访问。这些应用包括在线购物网站的大量商品图片的存储、社交网站的大量照片的存储等。这些应用不仅需要高性能的图片访问，而且图片数量也非常庞大，远远超过 GFS 的设计目标。不过，这些应用对图片的访问方式却比较简单，只需要根据一个唯一标识符取出图片或存入图片，而不需要文件系统提供“在任意位置读写任意长度的数据”这样的操作。为满足这些特定应用的特定需求，Facebook 和淘宝公司都开发了自己的、用于图片存储的专用文件系统，如 Haystack[45]和 TFS[46]。这些文件系统与专门的应用有很高的耦合度，定制性强，文件系统的很多操作接口都不支持。它们虽然在特定的访问模式下的性能非常高，但适应性差。

5.3 负载特征和性能评价

数据中心的负载是多种服务，每个服务为互联网上数以亿计的用户提供在线服务，同时为改善服务质量，需要对海量用户数据进行离线分析和挖掘。在线服务的同时在线用户数量在数百万到数千万量级，要求每个读写请求的响应时间要尽可能少，而且并发访问的吞吐率要尽可能高。离线数据处理采用 MapReduce 进行大规模的并行计算，各个节点上运行的任务分别访问各自的数据，要求并发访问的聚合读写带宽要尽可能高。与网络文件系统不同的是，离线数据处理一个作业访问的数据量和并发度远比网络文件系统高，单个作业扫描的数据量通常是数 TB，甚至数 PB，并发度是数千甚至数万个节点。因此，分布式文件系统的性能评价标准包括读写请求的响应时间、并发访问的吞吐率和并发访问的聚合读写带宽。

6 展望：数据中心文件系统——高通量文件系统

互联网应用的蓬勃发展，极大地方便了人们的生活和工作。它们不仅改变着人们的生活习惯和观念，而且极大地推动着计算机技术本身的发展。随着云计算、虚拟化技术的发展，人们将越来越依赖于大型数据中心来获得各种服务、信息和资源，大量的计算和服务资源汇集到大型数据中心中。数据中心采用高通量计算机[47]和高通量文件系统。高通量文件系统将数据中心中大量的、低成本的存储资源有效地组织起来，服务于上层多种应用的数据存储需求和数据访问需求。近年来，数据中心的数据存储需求逐渐成为数据存储技术和文件系统发展的主要驱动力，高通量文件系统将成为一种重要的文件系统。

6.1 数据存储的需求特征

随着互联网服务数量和数据量的激增，数据中心的数量也在快速增长。据 2008 年的报道，谷歌已经有 36 个数据中心[48]，未来将发展成数百个数据中心[49]。据了解，我国的大型互联网公司也在建自己的数据中心。数据中心，特别是以谷歌、雅虎、Facebook、亚马逊（Amazon）、百度、腾讯、淘宝等大型互联网公司为代表的大型数据中心，在数据存储和数据访问方面有着与先前的应用非常不同需求特征。

第一，数据量非常庞大，而且增长速度很快。目前，一些大型互联网公司的数据规模已突破 10PB 量级，预计未来 5-10 年将突破 EB 量级（即 10^{18}）。例如，Facebook 的照片总量超过 20PB，平均每天上传的数据量在 8~9TB[45]。Facebook 的数据仓库总数据量超过 15PB，每天新增 60TB 的数据（压缩后 10TB）[50]。

第二，访问高度并发。互联网信息服务通常都有庞大的用户群，注册用户有数亿，在线用户有数千万。在线用户交互式地访问互联网提供的服务，要求任何时刻都能访问到他们的

数据。例如，2010 年淘宝网注册用户达到 3.7 亿，最多的时候每天 6000 万人访问淘宝网，平均每分钟出售 4.8 万件商品[51]。Facebook 目前有 7.5 亿注册用户[52]，每月的页面点击率（page view）是 2600 亿[53]，换算下来，平均每分钟的页面点击率超过 600 万。Facebook 的服务器每秒要服务 100 万个图片访问请求[45]。eBay 每天的页面点击率超过 20 亿次[54]。大量用户的并发访问造成大量的随机读写，对存储系统的访问延迟和数据高可用性带来很大的挑战。另外，对于离线进行的海量数据分析，一个 Hadoop 集群上同时有数万个 MapReduce 作业提交运行[52]，这也是必须应对的重要问题。

第三，文件数量巨大，大文件和小文件并存。目前一些互联网应用的图片数量已超过数百亿，预计未来 2-5 年将增长到数万亿。例如，Facebook 目前的图片数量超过 2600 亿，每周新增图片 10 亿张，大约为 60TB[45]。除了图片外，互联网上还有大量的音视频文件，数量也在迅速增长。例如，搜狐视频总量在去年就突破了 10 亿[55]。

第四，数据访问语义和访问接口不同于传统的文件系统。数据中心的文件系统是面向应用的，而不是面向终端用户的。终端用户通过访问互联网上的各种服务来获得信息或数据，真正与数据中心文件系统直接交互的是各种服务软件，如搜索引擎、云存储服务、万维网服务（web service）等。因此，数据中心的文件系统不需要传统文件系统中那些复杂的操作，例如 `link`（链接）、`rename`（重命名）等，也不需要复杂的目录结构。它们需要按某个全局唯一的关键字（key）来访问数据。数据可以是无结构的字符流，也可能是半结构化的键值对，或者是多维表。因此，数据中心所需要的是不同于传统文件系统的访问语义和访问接口。另外，数据中心中一个目录下的文件数量非常庞大，是传统文件系统无法支持的。

第五，数据共享与数据安全的保障越来越重要，而且与传统文件系统不同。数据中心支撑着多种应用，它们为用户提供各种不同的服务。数据中心中有些数据是需要被多种应用共享的。例如，谷歌的 gmail、gtalk、gdoc 等服务都共享用户信息。一群用户之间还可以共享文档、照片等。但在数据共享需求的同时，数据还有安全性需求。数据是不应被未授权用户或应用访问到的。数据中心的数据共享和数据安全的需求不同于传统文件系统的访问权限控制。比如，传统的文件系统采用基于会话的访问控制，一个会话从打开文件开始，到关闭文件为止。而这种会话方式对于数据中心的数据访问并不适用。

6.2 可能的技术创新

数据中心的文件系统要为数据中心运行的各种互联网在线服务，为大量的后台数据分析任务提供数据存储支持。由于数据量非常庞大，有效地控制数据存储的成本是数据中心不得不考虑的问题。而在性能方面，由于数据中心服务于大量高度并发的请求，它们更强调高吞吐率，而不是峰值读写带宽或单个读写请求的延迟[40]。这就是我们将数据中心的文件系统称为高通量文件系统的原因。因此，成本和性能是数据中心文件系统需要考虑的首要问题。数据中心并不单纯追求高性能，而是追求能否以更低的成本获得更高的性能，即高性价比[41]。

针对数据中心的数据规模、成本、性能等问题，谷歌公司在 20 世纪末本世纪初提出了一种解决方案[39]，通常被称为 Google 架构。其基本思想是：第一，硬件设施采用低成本的 PC 集群（与之对比的是昂贵的服务器集群）、普通的 IDE 磁盘（与之对比的是昂贵的高端磁盘阵列）和普通的以太网（与之对比的是昂贵的高性能通信网络如 Myrinet、Quadrics、InfiniBand 等）。第二，通过软件来解决系统规模的可扩展性、可靠性、高吞吐率和可用性等问题。按照这一架构，谷歌开发了它的文件系统 GFS[40]和它的并行数据处理系统 MapReduce[41]。GFS 也许可以被认为是第一个数据中心文件系统。GFS 采用集中式管理结

构，实践证明了这种结构有很好的扩展性，可以支撑谷歌由上万个结点构成的集群。

但是，随着数据中心过去 10 年的发展，GFS 在一些方面已经无法满足当前数据中心的需求了，它在设计上目前面临的几个主要挑战是：

第一，集中式元数据管理成为 GFS 的瓶颈。GFS 在最初设计时主要是针对搜索引擎这个应用的，搜索引擎需要将整个万维网（web）的网页都抓取回来以构建索引。虽然网页数据通常是大量的小文件，但是应用层将这些小文件聚合成了 GB 级以上的大文件。因此，GFS 设计目标是支持百万量级（甚至千万量级）的 GB 级的大文件[40]。但是数据中心发展到现在，其文件数量已经远远不止千万量级，而是已经到了百亿量级，2-3 年后将达到万亿量级。对于文件数量如此庞大的文件系统，GFS 那种依赖单个元数据服务器的内存来管理整个文件系统的所有元数据信息的集中式处理已经显得不合时宜了，谷歌新一代的文件系统（称为 Colossus）已采用分布式元数据管理[49, 56]。

第二，小文件的高度并发访问性能低。目前数据中心提供的很多服务，不仅需要大量的大文件访问，而且更有大量的小文件并发访问。大量的交互式用户产生大量的、高度并发的小文件访问，如网上购物的商品图片、GIS 系统的卫星图片、邮件、文档等。这类应用不仅需要文件系统能够提供小文件访问的高吞吐率以满足高并发的需求，而且还需要每个小文件访问的延迟很低以满足交互式用户的需求。而由于这些文件大多不超过 1MB，不仅 GFS 而且传统文件系统在访问一个文件时都需要多次磁盘读写和多次的网络来回，因此难于满足低延迟的需求。

第三，存储成本仍然很高。GFS 的存储成本主要来自它的容错机制。由于 GFS 是运行于低成本的、可靠性较低的服务器、网络、磁盘上，所以通过采用多复本来保障数据的可靠性和可用性。每个数据在 GFS 中一般保存 3 个复本。这导致存储成本变成 3 倍。如果数据中心需要存储 10PB 的数据，就需要装备 30PB 的存储设备。目前业界也采用了各种方法来降低存储的成本。比如，应用层采用将数据压缩后再存储[50]。另外，业界也在探讨更节省成本的冗余机制。比如，采用纠删码（erasure code）[57]或者采用多复本与纠删码相结合的方式[50]。

第四，应用与应用之间、用户与用户之间数据的隔离与安全保障仍有很多问题。计算资源和存储资源汇集到数据中心之后，众多的应用（服务）和用户将共享数据中心的计算资源和存储资源。但是，数据是具有归属性的，有些数据只能由特定用户访问，而有些数据则是公开的，任何人都可以访问。目前，数据的访问控制大多由各个应用（服务）自己负责，GFS 不负责数据的隔离性和安全性。但这种方式仍然有漏洞，比如黑客可以绕过应用来访问用户的数据。因此，数据中心文件系统需要在数据隔离性和安全性方面提供必要的支持。

第五，多个数据中心构成全局数据视图带来的问题有待解决。出于容灾和数据量庞大等原因，数据中心的数据需要存储在地域不同的多个数据中心中。互联网服务的一个特点就是其服务的用户群体是遍布世界各地的，而且用户还可能旅行到不同地方，因此，就近访问是很重要。GFS 设计之初仅针对一个数据中心内部的数据存储。在多数据中心下，有很多以前没有考虑的问题，包括数据如何放置、如何移动、如何保证数据一致性等，都需要认真对待。

综上所述，GFS 虽然是第一个数据中心文件系统，但是它在元数据管理、高并发小文件访问、存储成本、数据安全性和多数据中心支持等方面都存在局限性，不能满足当前数据中心的需求。而高通量文件系统需要通过在上述各个方面进行技术创新来克服 GFS 的这些局限，更好地满足数据中心应用的数据存储和数据访问需求。

7 我们过去的工作简介

中国科学院计算技术研究所的国家智能计算机研究开发中心（简称智能中心）在文件系统方面的工作可以追溯到 1992 年，当时智能中心全面展开了曙光 1000 超级计算机的研制工作。曙光 1000 在技术路线上采用当时并行计算机的主流架构——MPP 架构，计算节点基于英特尔的 i860 处理器。并行文件系统是 MPP 架构并行计算机的重要系统软件，1992 年到 1994 年，智能中心研制开发了曙光 1000 的并行文件系统 PFS[58]。之后，智能中心面向高性能计算应用的数据读写需求，先后为曙光系列高性能计算机（从曙光 2000 到曙光 6000）研制了多个版本的机群文件系统，包括 COSMOS、DCFS1、DCFS2、LionFS、DCFS3 和 HVFS。在研制这些文件系统的过程中，对于机群文件系统的一系列关键问题展开了研究，并且提出了相应的解决方案。

7.1 技术贡献

COSMOS[59]是智能中心在 1996 年到 2000 年期间为曙光 2000 和曙光 3000 超级服务器研制的机群文件系统。COSMOS 运行于基于 IBM Power3 的 AIX 机群上。它借鉴了当时加州大学伯克利分校 NOW 项目中的机群文件系统 xFS[69]的对等架构思想，每个机群节点都既是数据存储的服务器，又是文件访问的客户端。而且将名字空间和元数据管理从数据读写通路上分离出来。另外，COSMOS 还对客户端协作式缓存技术进行了探索。

从 2001 年起智能中心开始研制基于 Linux 机群的超级计算机，同时开始了面向 Linux 机群的机群文件系统的研制。2002 年研制出第一个原型系统 DCFS1。与 PVFS1 类似，它采用典型的 3 元架构：存储服务器、元数据服务器、文件访问客户端。一个文件的数据采用条带化的方式存储在多个存储服务器上。它与 PVFS1 不同之处包括：（1）对于元数据管理，它采用根子树划分方法来划分名字空间，并自动地将文件元数据分布到多个元数据服务器。（2）服务器端采用线程池、缓存，以及网络传输与磁盘读写重叠等多种机制提升大数据量访问的性能[60]。

DCFS2 是为曙光 4000A 超级服务器研制的，面向 PB 级存储的机群文件系统，其底层采用共享 IP-SAN 来存储数据。DCFS2 的贡献在于提出了解决海量元数据管理的分布式元数据处理技术、高效的 PB 级存储空间管理技术和提高大目录下文件名查找效率的技术。元数据按层次结构粒度自动地分布于多个元数据服务器上，并将两阶段提交协议与元数据处理协议结合起来，是真正意义上的分布式元数据处理[61]。DCFS2 采用变长分配单元与存储资源位图相结合的方式来管理 PB 级存储空间[62]，与其它基于 SAN 的共享文件系统相比，大大提升大存储空间的管理效率。DCFS2 在每个元数据服务器上采用扩展哈希（Extendible hashing）技术来组织和定位本机所管理的元数据[63]，使得大目录的访问效率大大高于传统的文件系统。

LionFS 主要是解决机群文件系统的高可扩展性，它采用对象存储接口来存储文件数据。元数据按照名字哈希自动地分布到多个元数据服务器上，并且采用两级元数据架构，将元数据的处理与元数据的存储分离。另外，它还利用客户端的隐式信息来进行服务器缓存、预取和文件级读写调度[64]，大大提升了多个并发读写访问流下的聚合读写带宽。

DCFS3 是为曙光 5000 超级计算机研制的、高可扩展、高可靠的机群文件系统，它仍然采用对象存储接口来存储数据。为充分利用曙光 5000 节点的大容量内存来提升元数据处理性能，它采用全内存元数据管理[65]、基于复制的元数据服务高可用机制[66]、基于复制的自动数据容错机制[67]，和自动容错的组件间数据传输机制。在曙光 5000 上使用 1055 个节点，

DCFS3 的峰值聚合读带宽达到 50GB/s，峰值聚合写带宽达到 30GB/s，文件创建吞吐率达到 2 万个/秒（文件长度为 0）。

HVFS 是为曙光 6000 超级计算机研制的、面向云计算中心负载的机群文件系统。云计算中心既要服务于数据密集型大规模科学计算应用，又要服务于企业级数据存储需求，甚至还要作为互联网服务的基础设施。因此，针对不同的应用负载，云计算中心需要不同类型的共享文件系统，包括高性能的并行文件系统、SAN 文件系统和分布式文件系统（如 HDFS）等。HVFS 将这些文件系统集成为一个虚拟的、更大的、全局文件系统。另外，HVFS 还针对大量并发小文件访问进行了优化，采用分布式扩展哈希（Distributed Extendible Hashing，DEH）来划分目录，采用一致性哈希（Consistent Hashing，CH）来分布元数据，将文件与其元数据一起存储，并自动聚合大量小文件，从而大大提高小文件访问的性能。

表 2 总结了我们过去在机群文件系统方向的工作，以及与相关系统的对比。

表2. 智能中心在文件系统方向的研究工作

我们的系统	主要特点	国际相类似系统
PFS (1992-1994)	并行读写接口； 条带化	Intel CFS (1989) [68]
COSMOS (1996-2000)	无专用的存储服务器； 条带组； 客户端协作式缓存	xFS(1995) [69]
DCFS1 (2001-2002)	多元数据服务器； 条带化； 服务器端优化	PVFS1（1996） [35] （单元数据服务器）
DCFS2 (2003-2004)	多元数据服务器； 元数据按层次结构的粒度分布； 共享 IP-SAN 设备； 条带化	GPFS（2002） [70] （对称式结构、无元数据服务器）
LionFS (2005-2006)	多元数据服务器； 元数据按名字哈希分布； 条带化； 多复本	Lustre（2002）[36] （单元数据服务器、条带化、利用高性能通信网络的底层接口（RDMA））
DCFS3 (2007-2008)	全内存元数据管理； 条带化； 多复本	GFS（2003）[40] （大粒度数据分布）
HVFS (2009-2011)	分布式元数据处理； 元数据分布采用 DEH 和 CH； 优化小文件访问性能	Colossus（2009）[49] （元数据采用 Bigtable、大粒度数据分布；）

7.2 对产业界的贡献

通过这些深入的、系统化的研究工作，我们不仅在文件系统方向积累了丰富的研究和开发经验，而且还将我们的科研成果和培养的学生向产业界辐射。北京龙存科技有限公司的创始人、CEO 唐荣锋曾是智能中心文件系统小组的博士生。他在文件系统小组学习的 5 年中，

先后参加了 DCFS2、ClusterNFS 和 LionFS 的研制和开发。作为主力研发人员，他为这些文件系统的设计和开发做出了巨大的贡献。同时他在参加这些项目开发的过程中，积累了丰富的文件系统开发经验。2006 年开始，国内很多应用都对海量数据的高效存取提出了迫切的需求，机群文件系统是最有前景的解决方案。于是，他抓住时机于 2007 年出去创业，创立了北京龙存科技有限公司。经过 3 年多的发展，龙存公司已经发展成为业界知名的存储公司，资产已超过亿元。

我们对产业界的另一个贡献，是将我们的科研成果 LionFS 通过技术转移的形式转移给曙光公司。曙光公司以 LionFS 为基础，对它进行了二次开发，形成了自己的存储系统产品——ParaStor200。目前，曙光的 ParaStor200 已经安装在深圳超算中心的曙光 6000 超级计算机上，高效地管理曙光 6000 的 16PB 存储，为多种类型的应用提供高性能的读写访问。

智能中心的文件系统小组还培养出了一批优秀的学生，在计算所获得硕士或博士学位后，他们有的在大学或科研院所继续从事科研工作，有的去企业从事产品开发工作，有的自己去创业。表 3 给出了文件系统小组培养的部分优秀毕业生，他们在各自的领域对计算机事业做着贡献。

表3. 文件系统小组培养的部分优秀毕业生

姓名	学位 （毕业年份）	参与的 研发项目	离开智能中心文件系统组 后的工作单位	现在的工作单位
王建勇	博士（1999）	COSMOS	伊利诺伊大学香槟分校 博士后	清华大学计算机系 副教授
朱宁宁	员工（1999）	COSMOS	在纽约州立大学石溪分校 计算机系获得博士学位	谷歌
张迟	硕士（1999）	COSMOS	在普林斯顿大学计算机系 获得博士学位	谷歌
贺劲	博士（2002）	DCFS1	Cluster File Systems 公司、 计算所、EMC	自己创业
吴思宁	博士（2004）	DCFS1	英国克兰菲尔德（Cranfield） 大学博士后、肯特（Kent） 大学博士后	Xyratex 公司
吕毅	博士（2004）	DCFS1	中国科学院软件所博士后	中国科学院软件所副研 究员
唐荣锋	（2007）	DCFS2、 ClusterNFS LionFS	创立北京龙存科技 有限公司	北京龙存科技有限公司 CEO
陈欢	博士（2009）	ClusterNFS		EMC
邢晶	博士（2010）	LionFS DCFS3		中国科学院计算技术 研究所助理研究员

8 总结

我们用表 4 来总结文件系统的发展脉络。表 4 从技术背景、负载特征、创新技术、性能

评价标准等方面对比了各种文件系统。

在当今这个信息化时代，数据成为推动计算机科学快速发展的重要因素。随着新应用的蓬勃发展和数据量的激增，有很多问题亟待解决。在这篇文章中，我们粗略回顾了文件系统的发展历程，并且粗浅地探讨了未来的数据中心文件系统可能的技术创新。期望借此与广大同行交流。

表4. 文件系统的发展脉络

阶段	产生的技术背景	负载特征	典型代表	主要的创新技术	性能评价标准
单机文件系统	分时操作系统 多用户共享磁盘	多用户并发访问 多进程并发访问	Unix FS FFS LFS JFS WAFL XFS ZFS	树型目录结构 索引节点（i-node） 流式访问接口 柱面组 元数据修改日志 B+树组织 写时复制 存储池	读写请求响应时间 聚合读写带宽
网络文件系统	局域网 TCP/IP 协议 RAID 光纤通道网络	多客户端共享访问 多用户共享访问	NFS AFS NAS SAN 文件系统	XDR RPC VFS 无状态服务器 多服务器结构	聚合读写带宽
并行文件系统	MPP 超级计算机 高性能互连网络 并行编程	一个作业的多任务对同一文件不现位置的并行访问 一个读写请求的并行处理	Concurrent File System Vesta PVFS Lustre	文件的条带化存储 并行读写接口 元数据管理与数据存储分离	并行读写带宽
分布式文件系统	搜索引擎 互联网服务 Google 架构 大规模 PC 集群	数千万在线并发访问 数万并发大粒度访问	GoogleFS HDFS Haystack TFS	非 POSIX 接口和语义 集中管理、分散存储 全内存元数据处理 多个复本	读写请求响应时间 并发访问吞吐率 聚合读写带宽

参考文献：

[1] P. Hansen, “The evolution of operating systems”, in Classic Operating Systems: From Batch Processing to Distributed Systems, P. Brinch Hansen, Ed. Copyright 2000, Springer-Verlag, New York.

[2] F. Corbató and V. Vyssotsky, "Introduction and overview of the Multics system", *AFIPS Fall Joint Computer Conference*, 1965

[3] R. Daley and P. Neumann, "A general-purpose file system for secondary storage" *AFIPS Fall Joint Computer Conference*, 1965

[4] D. Ritchie, "The Evolution of the Unix Time-sharing System", *AT&T Bell Laboratories Technical Journal*, Vol.63, No.6 Part 2,October 1984

[5] D. Ritchie and K. Thompson, "The Unix time-sharing System", *ACM Symposium on Operating Systems Principles(SOSP)*, October, 1973

[6] M. McKusick, W. Joy, S. Leffler, R. Fabry, "A Fast File System for UNIX", *ACM Transactions on Computer Systems*, Vol. 2, No. 3, August 1984, Pages 181-197.

[7] M. Rosenblum and J. Ousterhout, "The Design and Implementation of a Log-Structured File System", *ACM Transactions on Computer Systems* **10**(1), February 1992, pages 26-52

[8] S. Tweedie, "Journaling the Linux ext2fs Filesystem", *The Fourth Annual Linux Expo*, 1998

[9] A. Sweeney, D. Doucette, W. Hu, C. Anderson, M. Nishimoto, and G. Peck, "Scalability in the XFS File System", *Proceedings of the Winter 1996 USENIX Conference*, January 1996, pages 33-44

[10] J. Bonwick, M. Ahrens, V. Henson, M. Maybee, M. Shellenbaum, "The Zettabyte File System（ZFS）", 2002.

[11] D. Patterson, G. Gibson and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)", *Proceedings of the International Conference on Management of Data (SIGMOD)*, June 1988

[12] http://en.wikipedia.org/wiki/TCP/IP_model

[13] http://en.wikipedia.org/wiki/ARPANET

[14] E. Taft and R. Metcalfe, "Pup Specifications", Inter-Office Memorandum of Xerox Parc, Palo Alto, June, 1978 and October, 1975

[15] D. Brown, J. Burchfiel, D. Murphy, R. Tomlinson, "TENEX, a Page Time Sharing System for PDP-10", *Communications of the ACM*, Vol. 15, No. 3, March 1972, pp. 135-143

[16] "Interim File System Overview", Inter-Office Memorandum of Xerox Parc, Palo Alto, Nov., 1977

[17] J. Mitchell and J. Dion, "A Comparison of Two Network Based File Servers*", Communications of the ACM*, Vol. 25, No. 4, April 1982, pp. 233-245

[18] M.Satyanarayanan, "A Survey of Distributed File Systems", *Annual Review of Computer Science 1990*

[19] D. Brownbridge, L. Marshall, B. Randell, "The Newcastle Connection or Unixes of the World Unite", in Classic Operating System, Springer-Verlag New York, Inc., New York, 2000, pp. 528-549

[20] D. Nowitz, "UUCP Administration", UNIX Research System Papers, Tenth Edition, Vol. II, Saunders College Publish, 1990, pp. 563-580

[21] J. Postel and J. Reynolds, "The File Transfer Protocol", RFC 959, Oct. 1985

[22] R.Sandberg, D. Goldberg, R. Kleiman, D. Walsh, and B. Lyon, "The design and implementation of the Sun Network Filesystem", *Proceedings of the Summer 1985 USENIX Technical Conference*, Jun, 1985, pp. 119-130,

[23] M. Satyanarayanan, J. Howard, D. Nichols, R. Sidebotham, A. Spector, M. West, "The ICT Distributed File System: Principles and Design", *SOSP'85*, 1985, pp. 35-50

[24] "NFS: Network File System Protocol Specification", RFC 1094, Sun Microsystems, Inc. March 1989

[25] "XDR: External Data Representation Standard", RFC 1014, Sun Microsystems, Inc. June 1987

[26] "RPC: Remote Procedure Call Protocol Specification", RFC 1057, Sun Microsystems, Inc. June 1988

[27] U. Vahalia, “*UNIX 高级教程：系统技术内幕*”，聊鸿斌等译，清华大学出版社，1999

[28] R. Morris, “The Evolution of Storage Systems”, IBM System Journal, 2003.

[29] Wikipedia. http://en.wikipedia.org/wiki/Network-attached_storage

[30] “A Storage Architecture Guide”, Auspex Technology Report, Auspex Systems, Inc. 2001.

[31] P. Dibble, M. Scott, C. Ellis, “Bridge: A High-Performance File System for Parallel Processors”, ICDCS’88

[32] J. French, T. Pratt, M. Das, “Performance Measurement of a Parallel Input/Output System for the Intel iPSC/2 Hypercube”,SIGMETRICS’91

[33] R. Bordawekar, A. Choudhary, et al., “An Experimental Performance Evaluation of Touchstone Delta Concurrent File System”, ICS 93

[34] P. Corbett and D. Feitelson, "The Vesta Parallel File System", *ACM Transactions on Computer Systems* 14(3), August 1996, pages 225-264

[35] P. Carns, W. Ligon III, R. Ross, R. Thakur, “PVFS: A Parallel File System for Linux Clusters”, *Proceedings of the 4th Annual Linux Showcase and Conference*, October 2000, pp. 317-327

[36] P. Braam, “Lustre® File System”, White Paper, Cluster File Systems, Inc., July 2007

[37] S. Adee, “37 Years of Moore’s Law”, *IEEE Spectrum*, May 2008

[38] P. Corbett, D. Feitelson, S. Fineberg, Y. Hsu, B.Nitzberg, J. Prost, M. Snir, B. Traversat, P. Wong, “"Overview of the MPI-IO parallel I/O interface”, *IPPS’95 Workshop on Input/Output in Parallel and Distributed Systems*, April 1995

[39] L. Barroso, J. Dean, U. Hölzle, “WEB Search for a Planet: The Google Cluster Architecture”, *IEEE Micro*, March-April, 2003

[40] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google File System”, *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP’03)*, 2003, pp. 29-43

[41] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” in *OSDI 2004*, pp. 137–150

[42] The Hadoop project, *http://hadoop.apache.org/*

[43] D. Fetterly, M. Haridasan, and M. Isard, “TidyFS: A Simple and Small Distributed File System”, *2011 USENIX Annual Technical Conference*, June, 2011

[44] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, “Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks”, *European Conference on Computer Systems (EuroSys)*, Lisbon, Portugal, March 21-23, 2007

[45] D. Beaver, S. Kumar, H.C. Li, J. Sobel, P. Vajgel, “Finding a needle in Haystack: Facebook’s photo storage”, *OSDI’10*

[46] TFS 介绍，http://code.taobao.org/trac/tfs/wiki/intro

[47] 詹剑锋，王磊，孙凝晖，“高通量计算机的性能评价”，*中国计算机学会通讯*，第 7 卷， 第 7 期，2011 年 7 月

[48] Map of All Google Data Center Locations, http://royal.pingdom.com/2008/04/11/map-of-all-google-data-center-locations/

[49] J. Dean, “Designs, Lessons and Advice from Building Large Distributed Systems”, Keynote speak at

The 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systmes and Middleware (LADIS 2009), Oct. 2009 http://www.cs.cornell.edu/projects/ladis2009/program.htm#keynote3

[50] A. Thusoo, S. Anthony, N. Jain, R. Murthy, Z. Shao, D. Borthakur, J. S. Sarma, H. Liu, "Data Warehousing and Analytics Infrastructure at Facebook", *SIGMOD'10*, June 2010

[51] 淘宝网单日交易额达到 19.5 亿，2011 年 1 月 7 日，http://it.sohu.com/20110107/n278722475.shtml

[52] Facebook user count reaches an astounding 750 million, June 27, 2011 http://ssdigitalmedia.com/2011/06/27/facebook-user-count-reaches-an-astounding-750-million/

[53] Facebook boasts 11 times more page views than My Space, 59 times more than Twitter, Jan 5, 2010 http://royal.pingdom.com/2010/01/05/facebook-twitter-myspace-page-views/

[54] Rich Miller, eBay Right-Sizing Infrastructure As Growth Surges, May 27, 2011 http://www.datacenterknowledge.com/archives/2011/05/27/ebay-right-sizing-infrastructure-as-growth-surges/

[55] 薛蓓，2010 年六月搜狐视频总量突破 10 亿大关， 2010 年 8 月 12 日，http://news.ccidnet.com/art/1032/20100812/2151775_1.html

[56] K. McKusick and S. Quinlan, "GFS：Evolution on Fast-Forward", *ACM Queue*, August 7, 2009; also found in *Communications of the ACM*, Vol.53, No. 3, March 2010, pp.42-49

[57] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. Replication: A Quantitative Comparison", *the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, March 2002

[58] 孙凝晖，"并行文件系统的设计"，*计算机学报*，第 17 卷，第 12 期， 1994 年 12 月，P. 938-945

[59] 王建勇，祝明发，"可扩展单一映象文件系统的设计、实现及评价"，《*计算机研究与发展*》，1999

[60] J. Xiong, S. Wu, D. Meng, N. Sun, G. Li, "Design and Performance of the Dawning Cluster File System", The Proceedings of 2*003 IEEE International Conference on Cluster Computing(Cluster2003)*, Dec. 2003, pp. 232-239

[61] J. Xiong, R. Tang, S. Wu, D. Meng, N. Sun, " An Efficient Metadata Distribution Policy for Cluster File Systems", *2005 IEEE International Conference on Cluster Computing (Cluster2005)*, Sept. 2005

[62] J. Xiong, R. Tang, Z. Fan, J. Ma, H. Li, D. Meng, N. Sun, G. Li, "A Storage Space Management Policy for a Cluster File System", *The 8th International Conference on High-Performance Computing in Asia-Pacific Region (HPC Asia 2005)*, 2005, pp.240-245

[63] R. Tang, D. Meng, S. Wu, "Optimized Implementation of Extendible Hashing To Support Large File System Directory", *2003 IEEE International Conference on Cluster Computing (Cluster2003)*, Dec. 2003

[64] H. Chen, J. Xiong and N. Sun, "A Novel Hint-based I/O Mechanism for Centralized File Server of Cluster", *2008 IEEE International Conference on Cluster Computing (Cluster2008)*, Sept. 2008

[65] J. Xing, J. Xiong, J. Ma and N. Sun, "Memory based metadata server for cluster file systems ", *2008 Seventh International Conference on Grid and Cooperative Computing (GCC2008)*, Oct. 2008

[66] Z. Chen, J. Xiong, D. Meng, "Replication-based Highly Available Metadata Management for Cluster File Systems", *IEEE International Conference on Cluster Computing 2010 (IEEE Cluster 2010)*, Sept. 2010

[67] L. Cao, Y. Wang, J. Xiong, "Building Highly Available Cluster File System Based on Replication", *The Tenth International Conference on Parallel and Distributed Computing, Applications and*

Technologies (PDCAT'09), Dec. 2009

[68] P. Pierce, "A concurrent file system for a highly parallel mass storage subsystem", *1989 Hypercube Concurrent Computers and Applications Conference*

[69] T. Anderson, M. Dahlin, J. Neefe, D. Patterson, D. Roselli, and R. Wang, "Serverless Network File Systems," *ACM Transactions on Computer Systems* 14(1), Feb. 1996, pp. 41-79

[70] F. Schmeck, R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters", *First USENIX Conference on File and Storage Technologies(FAST2002)*, Jan. 2002

作者简介：

熊　劲：　中国科学院计算技术研究所先进计算机系统实验室副研究员，xj@ncic.ac.cn
潘锋烽：　中国科学院计算技术研究所先进计算机系统实验室硕士研究生
刘凯捷：　中国科学院计算技术研究所先进计算机系统实验室硕士研究生
张子刚：　中国科学院计算技术研究所先进计算机系统实验室硕士研究生
马久跃：　中国科学院计算技术研究所先进计算机系统实验室博士研究生
姜　继：　中国科学院计算技术研究所先进计算机系统实验室硕士研究生
孙凝晖：　中国科学院计算技术研究所所长、研究员